Policy Enforcement in Stub Autonomous Domains

Gene Tsudik

IBM Research Laboratory, CH-8803 Rüschlikon, Switzerland, *gts@zurich.ibm.com*

Abstract. Interconnection across administrative boundaries prompts the need for comprehensive policy enforcement (i.e., access control) with respect to inter-domain packet traffic. Due to the nature of the communication services they provide, stub and transit domains require different mechanisms for policing inter-domain traffic. This paper addresses the design of a policy enforcement mechanism geared specifically towards stub domains. With the aid of some basic concepts borrowed from *Visa* protocol[5], a much more powerful mechanism is developed and analyzed. Protocol implementation and experimental results are discussed.

Keywords: inter-domain communication, authentication protocols, data integrity, communication security, network protocols, internetworking.

1 Introduction

Increasing use of computers for communication has prompted widespread interconnection of autonomous networks. In an environment of interconnected Administrative Domains (ADs), access to network resources is an issue of growing concern. In the absence of special mechanisms, network interconnection using existing internetworking protocols (e.g., IP [21] or OSI [10]) attempts to achieve full connectivity. However, ADs should be able to interconnect without exposing their internal resources to unrestricted external access. Moreover, internetwork components should be able to control incoming and outgoing traffic by specifying or constraining the ADs to, and through, which the traffic can flow [8].

While complete autonomy implies no interconnection, increased *"openness"* sacrifices autonomy. Thus, each participant organization must reach its own tradeoff between autonomy and interdependence. The particulars of this tradeoff are embodied in what we refer to as **policy**. Consequently, *policy enforcement* refers to the application of policy to internetwork communication.

A framework for internetwork policy enforcement is presented in [7] and a complete architecture based on this framework is described in [28]. This paper is concerned only with policy enforcement with respect to non-transit (i.e., stub) internetwork traffic. In other words, the issue at hand is how an AD can control both inbound (addressed to a local end-system) and outbound (sourced by a local end-system) traffic at its network boundaries.

According to the framework presented in [7], the goals of a stub policy enforcement protocol are:

---

* This work was performed while the author was affiliated with the Computer Networks and Distributed Systems Laboratory at the University of Southern California.

1. Flexibility - protocol operation must be almost entirely dependent on the particular access control and authentication policies of the participating AD.
2. Functionality - the protocol must provide protection against: i) unauthorized AD entry/exit, and ii) modification, substitution and replay of legitimate inter-AD traffic.
3. Layering - the entire protocol must be situated at the network layer. (The reader is referred to [7, 28] for the discussion of the motivating factors leading to this requirement.)
4. Efficiency - the protocol overhead must be low enough so as not to affect the configuration of existing higher-layer protocols and applications. Moreover, there must be no interference with intra-AD communication and, similarly, no impact upon internal resources not involved in communication with the outside.

1.1 *Visa* Protocol Overview

In brief, *Visa* protocol is a mechanism for controlling the flow of packet traffic to and from end-systems in a stub AD. Before an end-system can communicate across its AD boundary, the communication has to be authorized according to the policies of both source and destination ADs. Authorization can be obtained via a dialog with an Access Control Server (ACS) on local and destination ADs. The need for and particulars of this dialog are determined independently by the administration of each AD involved. When the communication is approved by both end-point ADs, the respective ACSs issue *visas* to the requesting end-system.

A *visa* is a cryptographically sealed certificate issued by an Access Control Server (ACS). It contains, among other things, a secret quantity, known as the *visa-key*. Each packet belonging to an authorized stream carries a *visa-stamp*, which indicates that the packet is allowed to leave (or enter) an AD's network. A visa-stamp is a function of the visa-key and the packet's data. It is attached to the packet by the originating end-system and subsequently re-computed and verified by the two border routers of the respective end-point ADs.

In *Visa* protocol, border routers are not solely responsible for making access control decisions. By issuing a visa, a *higher*, more intelligent authority (i.e., an ACS) has pre-computed a decision such as *"end-systems H_a and H_b are allowed to communicate"*, or *"end-system H_a can be trusted to pay its bills"*. The task of a router is thus reduced to ensuring that a visa is valid and is being used correctly; the expensive part of the policy enforcement is done once per connection, by the ACSs of the end-point ADs, rather than once per packet, by the border routers.

1.2 History

Previous work, in particular [5], resulted in the development of two *Visa* protocol models based on different philosophies with regard to state in visa-routers. The original *stateful* model requires that participating border routers maintain **reliable** tables of active visas. In it, ACSs explicitly distribute visas to visa-routers.

Although the loss of state in a visa-router is not fatal to communication, overhead is incurred in the process of re-establishing the necessary state. In contrast, the *stateless* model avoids the necessity for the distributed state, but requires some additional encryption steps. The stateless model has several advantages: higher fault tolerance (insofar as routers), lower router storage requirements, and the ability to accommodate multiple visa-routers without additional overhead. All this is gained at the price of higher per-packet processing costs and increased packet size.

In the remainder of this paper, we use the experience from previous work to develop a new *Visa* protocol suitable for general-purpose policy enforcement in a stub AD environment. In the next section, the goals of *Visa* protocol are formalized. Network environment is discussed in Section 3. *Visa* protocol participants and their respective requirements are addressed in Section 4. Section 5 presents the new *Visa* protocol, and Section 6 addresses the key design issues and choices. Section 7 analyzes the security of *Visa* protocol and Section 8 evaluates the storage requirements. Protocol implementation details and performance results are presented in the appendix.

2 Goals

The primary goal of *Visa* protocol is to allow an AD to control communication between its constituent end-systems and end-systems in other ADs. If the end-systems involved can be trusted, then a stronger goal can be met: we can control the transmission of packets to and from *a specific* end-system in another AD. In a datagram network, as opposed to a circuit-switched network, the only information available about a packet must be attached to the packet rather than inferred from the route the packet follows. Therefore, we can state these goals more directly as follows.

1. A packet can leave the source AD, AD_{src} if and only if ACS_{src} has authorized the source end-system, H_{src} to communicate with the destination end-system, H_{dst}.
2. A packet can leave AD_{src} if and only if it originated at H_{src} within a reasonable time interval, has not been modified in transit and is addressed for H_{dst}.
3. A packet can enter the destination AD, AD_{dst} if and only if ACS_{dst} has authorized H_{src} to communicate with H_{dst}.
4. A packet can enter AD_{dst} if and only if it originated at H_{src} within a reasonable time interval, has not been modified in transit and is addressed for H_{dst}.

Another fundamental goal is not to impact intra-AD communication, nor to impose additional security measures upon unequipped end-systems, i.e., those that do not participate in inter-AD communication. Similarly, we wish to limit the overhead imposed upon ADs that are not concerned with controlling external access.

Finally, we would like to minimize the costs imposed by *Visa* protocol, including:

- Additional per-packet processing in border routers and end-systems
- Storage requirements for routers and end-systems
- Extra communication during connection setup
- Additional packet length (additional length increases latency and decreases throughput)
- Cost of recovery from router crashes

3 Network Environment

We assume that the internetwork closely follows the model of the DARPA Internet [20], which is substantially similar to the Open Systems Interconnection (OSI) model [10]. The essential features of this environment are:

- End-systems are autonomous and cannot necessarily be trusted.
- ADs are interconnected with routers; between any pair of end-systems in different ADs there are at least two routers, one belonging to each of the ADs. Conceptually, the connection between two ADs is a pair of half-routers connected via a trusted link. Each half-router can be trusted by its own AD but not necessarily by any other AD. The terms *border router* and *inter-AD router* are equivalent.
- All information flows via datagram packets. A packet consists of a *header* that includes addressing and other control information, and a data segment that is not intelligible to routers.
- A packet may flow through several *untrusted* ADs on its way to the destination.
- End-system addresses, both source and destination, can be forged. It is not possible (using hardware methods) to determine reliably which end-system actually sent a packet or to prevent a packet from being seen by unauthorized end-system.
- Packets traveling across an internetwork may be: i) lost, ii) duplicated, and iii) re-ordered.
- Successive packets between a given end-system pair may travel along different routes.

Lastly, there must exist a global name service which, in a secure and reliable fashion, provides a mapping from end-system network addresses to AD identifiers in addition to the more traditional mapping between end-system names and addresses. Along with AD identifiers, the name service has to provide:

- Addresses of ACS-s within an AD.
- Public key certificates for a given AD (or an ACS assuming each ACS is assigned a distinct certificate)

4 Participants

Visa protocol involves three types of participants: access control servers, border routers, and end-systems. These participants and their responsibilities are described in this section.

4.1 ACSs

An ACS is an end-system, usually dedicated for security reasons, that is primarily concerned with access control. Each AD that implements *Visa* protocol has at least one ACS, responsible for authorizing its constituent end-systems for communication with end-systems in other ADs.[2] Multiple ACSs may be necessary for availability and performance reasons.

Each ACS knows of a number of local border routers that implement *Visa* protocol. ACSs are trusted and are sufficiently secure to defend against hostile attacks. The security of the overall protocol requires that ACSs be secure and that they employ an authenticated and secure channel for communication with local end-systems and routers. Furthermore, each ACS must be *identifiable* by a unique public key pair $[\, EK_{ACS}, DK_{ACS} \,]$ where EK_{ACS} is the ACS's public (encryption) key, and DK_{ACS} is the corresponding secret (decryption) key. Also, each AD is assumed to be a participant in a global, internet-wide certification scheme, whereby each AD (or each ACS therein) has a public-key certificate, $CERT_{ACS}$, issued by a well-known certification authority. Each ACS certificate contains (among other fields): ACS's address, EK_{ACS}, the name of the issuing authority and the certificate signature. This signature is computed with the issuing authority's private key, hence, anyone in possession of the corresponding public key can verify the certificate's validity and thus authenticate the certificate holder.

4.2 Border Routers

A border router is an end-system dedicated (for reasons of performance and security) to packet forwarding. Routers that use *Visa* protocol to enforce access controls are called visa-routers. All inter-AD connections must be implemented via visa-routers. Each visa-router knows the ACSs in its AD, is willing to accept visas issued by some or all of these ACSs, and trusts their decisions about authorizing and terminating connections.

A visa-router allows any external party to communicate with any registered, internal ACS. Similarly, visa-routers allow all registered, local ACSs to communicate with any external party. Such trust is reasonable because ACSs are assumed to implement sufficient defense mechanisms and to enforce AD's policy.

[2] If a participant AD does not have an ACS, its end-systems will still be able to communicate with the end-systems in other ADs, although the AD in question will be subject to risks associated with the uncontrolled access.

However, this does not imply that visa-routers should let the ACS-originated outbound traffic go unchecked. In order to detect *bogus* ACS packets and prevent replay of pre-recorded ACS-sourced packets, a visa-router must verify packet signatures and validate packet timestamps of all packets purportedly originating at a local ACS.

It is more difficult to control traffic coming in from the outside destined for a local ACS. Because such traffic can originate anywhere in the internetwork its validation would require a visa-router to have means for verifying signatures and timestamps generated by a possibly large number of sources. This would necessitate a lot of state information in visa-routers which is clearly undesirable and impractical for reasons of performance. The alternative is not to scrutinize in-bound ACS traffic and let the ACSs filter out fraudulent packets. Visa-routers can still maintain some control by making sure that local ACSs do not get *flooded* by in-bound traffic. The disadvantage of this approach is that it may violate one of our fundamental goals of not allowing *unauthorized* externally-sourced traffic consume internal network resources. This subject is addressed further in Section 6.

Assuming that each AD employs visa-routers, each inter-AD packet travels through at least two such routers. A visa-router must scrutinize every packet it receives; packets without visas cannot be forwarded (except for those to or from trusted entities of the router's own AD). In section 5.1 we describe a mechanism for a visa-router to inform an end-system that a visa is required for inter-AD communication.

Packets originating at or destined for unequipped end-systems are discarded by the visa-routers since these end-systems are (by definition) not allowed any external access.

If the two stub ADs are not directly connected, packets will pass through the routers of transit networks. Visa-routers within a single transit AD are assumed to trust each other, and transfer transit packets via secure channels to prevent unauthorized entrance or exit. Non-visa routers in transit ADs treat visa-stamped packets as regular internet packets.

4.3 Participating End-systems

An end-system attempting communication outside of its AD must be able to obtain a visa allowing it exit from the local AD and entry to the destination AD. In order to obtain exit authorization it needs to contact one of the local ACSs which (upon authenticating the requesting end-system and checking its access control lists) then contacts an ACS in the destination AD and requests entry authorization on behalf of the original end-system. After establishing entry authorization, the remote ACS issues a visa which is subsequently delivered to the requesting end-system. Thereafter, a visa-stamp (computed with the corresponding visa-key) must be attached to every packet sent from the requesting end-system to the apparent destination.

An end-system, unlike a visa-router, does not have to have reliable knowledge of the local ACS's address; this may instead be supplied by a visa-router when

an end-system first attempts to communicate across the AD boundary (see section 5.1). An end-system may still need to use an authentication protocol (e.g., Kerberos [25]) to make sure it is really talking to the ACS.

Since packet reception is a passive operation, the destination end-system is not required to initiate any actions. Of course, in most types of communication, packets flow in both directions, so each end-system is both a source and a destination. Therefore, to avoid additional overhead we assume that an AD may allow its ACS to issue *two-way* visas automatically if no authentication of the remote destination is required.

In summary, the requirements for a participating end-system are as follows:

- an authentication mechanism to allow for a dialog with a local ACS
- secure storage for active visas
- a means for generating visa-stamps

Lastly, a participating end-system must be identifiable, i.e., it must be assigned a unique key or key-pair (depending upon the encryption method).

It should be noted that *Visa* protocol, by itself, does not provide for multi-level security, nor does it eliminate a variety of covert channels. In the absence of additional non-discretionary controls, a participating end-system may still compromise access controls by serving as a conduit for communications between unauthorized end-systems.[3]

5 Protocol

Visa protocol consists of three phases. In the setup phase, an end-system obtains authorization for exiting its own AD and entering the destination AD. If successful, it culminates with the issuance and distribution of visas to all principals involved. In the packet forwarding phase, the visa-key is used to generate packet data signatures that are attached to all packets belonging to an authorized connection. Finally, the teardown phase involves the termination of a visa either because of normal expiration or by explicit revocation. In the remainder of this section, each protocol phase is discussed separately.

5.1 Setup Phase

The purpose of the setup phase is to i) authorize communication between two end-systems by the ACSs in the respective ADs, ii) issue a visa which embodies this authorization, and, iii) distribute it to all parties involved. The placement of the protocol participants is illustrated in Figure 1.

[3] See section 6.

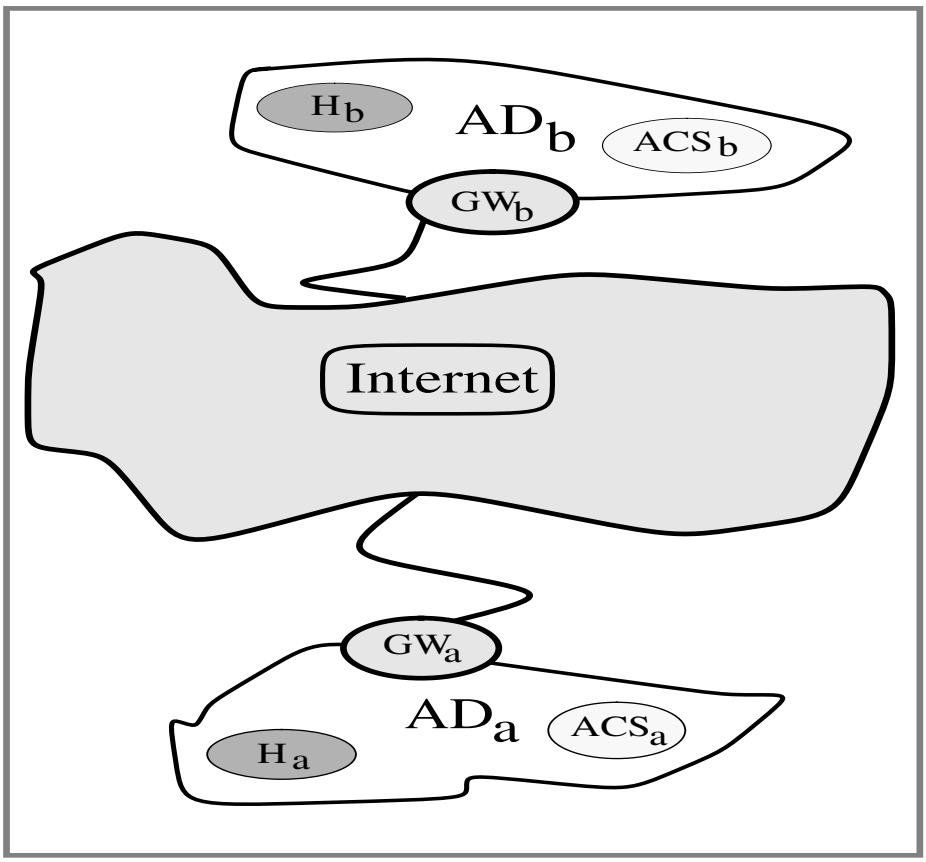

Fig. 1. Two ADs participating in *Visa* protocol

Exit Authorization The protocol is put in motion when an end-system, H_a, in AD_a begins communication with another end-system, H_b, in a different AD, AD_b. H_a may already know that its intended destination is in a different AD, either because it has previously communicated with H_b or it may have discovered this through some external mechanism (e.g., a name server). If so, H_a may communicate directly with an ACS in its AD, ACS_a. Otherwise, it may discover that H_b is in a different AD when its packet reaches the exit visa-router. Since the packet carries no visa-stamp, the exit visa-router replies with a REDIRECT packet. REDIRECT is essentially a means of notifying H_a that the intended destination is non-local, and that it must *"apply"* for a visa with a local ACS.

$$REDIRECT = [\, H_a, H_b, ACS_a \,] \tag{1}$$

The protocol begins by H_a requesting authorization from ACS_a. It does so by sending a HOST-REQUEST packet to ACS_a.

$$HOST-REQUEST \ = \ [\ H_a, H_b, TS_{H_a}\ ]^{K_{H_a}} \tag{2}$$

To maintain data integrity, HOST-REQUEST is signed with H_a's key, K_{H_a}. If conventional encryption is used, K_{H_a} is a key known only to H_a and ACS_a. With public key encryption, K_{H_a} is the private (secret) key known only to H_a. A timestamp, TS_{H_a}, is included to demonstrate the timeliness of the packet to ACS_a. (Timeliness is, of course, relative to the value of Δ_T, the maximum allowed clock skew.)

Next, ACS_a has to authorize communication between H_a and H_b. This step is dependent on the particular policy employed by AD_a. For example, it may involve a higher-level authentication dialog between AD_a and H_a. The details of this procedure are beyond the scope of this discussion.

If and when exit authorization is established, ACS_a composes a VISA-REQUEST packet:

$$VISA-REQUEST \ = \ [\ H_a, H_b, TS_a\ ]^{DK_{ACS_a}} \tag{3}$$

The packet is signed with ACS_a's secret key and timestamped to help verify both data integrity and timeliness. The timestamp, TS_a, is guaranteed to be unique; thereafter, it is used as a visa identifier. As with HOST-REQUEST, the signature is not needed if a higher-level authentication dialog is used between ACS_a and its counterpart in AD_b. Nevertheless, our purpose is to allow authentication to take place at the earliest possible time rather than relying on the presence of a higher-layer AD-dependent mechanism.

In order to deliver a VISA-REQUEST, ACS_a has to locate its counterpart in the destination AD, AD_b. It may know the address of ACS_b because of previous communication in which case VISA-REQUEST may be sent directly. It could also obtain ACS_b's address via a name service query [15]. Alternatively, VISA-REQUEST may be sent addressed to H_b (the destination end-system). This is possible because visa-routers are required to reroute *all* VISA-REQUEST packets to one of the local ACSs.

Furthermore, in case of no previous communication with AD_b, ACS_a may choose to speed up the setup process a little by including its certificate, $CERT_{ACS_a}$ with the VISA-REQUEST packet. This would save ACS_b the need to request this certificate explicitly (from either ACS_a or some name server).

Entry Authorization When a VISA-REQUEST reaches AD_b, the intervening visa-router, GW_b, forwards it to one of the local ACSs, ACS_b.

First, ACS_b verifies that H_b indicated in the VISA-REQUEST is in fact an end-system in AD_b. This is necessary in order to minimize time spent on potentially malformed VISA-REQUESTs. Next, before proceeding to authorize the connection ACS_b has to validate the VISA-REQUEST. In order to authenticate its contents (i.e., re-compute the signature), ACS_b needs the public key of ACS_a, EK_{ACS_a}. To obtain this key ACS_b has to know the AD identifier of ACS_a (i.e., AD_a), which is not included in the VISA-REQUEST. However, ACS_b can query

its name service with H_a (which is part of VISA-REQUEST) and obtain AD_b, ACS_b and EK_{ACS_a}.

With the help of EK_{ACS_a} ACS_b can re-compute the signature of the VISA-REQUEST and verify both its origin and data integrity. Both timeliness and uniqueness of the VISA-REQUEST are inferred from the enclosed timestamp, TS_a (using, for example, the loosely-synchronized clock scheme developed by Liskov et al. at MIT[14]).

Next, ACS_b authorizes communication between H_a and H_b. As before, this step is dependent on AD_b's policy. At last, when communication is authorized, ACS_b issues a fresh visa in a form of a VISA packet.

$$VISA{-}GRANT = [H_a, H_b, auth{-}type, TS_a, Expiration, Conditions, (S_b^a)^{EK_{ACS_a}}]^{DK_{ACS_b}} \tag{4}$$

Since the structure of a *VISA-GRANT* packet is crucial to the security of the protocol, we now consider the individual packet fields in more detail:

- H_a, H_b are the two end-systems that are authorized to use this visa. In order to associate packets with a particular connection, visa-routers GW_a and GW_b, need these end-system addresses to locate the appropriate visa-key.
- *auth-type* denotes the signature method to be used for packet signature computation.
- S_b^a is the so-called *visa-key*. It is subsequently used to compute packet signatures for traffic flowing between H_a and H_b. Depending on the signature method, S_b^a may be an encryption key (as in DES-based MAC) or a secret prefix/suffix for use in conjunction with MD4. (See Appendix A). Because a visa-key has to be kept secret to prevent interception by potential intruders, it is encrypted with ACS_a public key.
- TS_a is the timestamp assigned to the visa by ACS_a and *approved* by ACS_b. It is used primarily as a *nonce* [18], i.e., a unique, hereto unused, visa identifier.
- Expiration indicates the condition(s) for the termination of a visa. May be expressed as any combination of:
  - maximum lifetime of a visa (e.g., in msec)
  - maximum inactivity period
  - maximum number of packets
  - maximum amount of data transferred (e.g., in Kbytes)
- The Conditions field can be used to express the visa usage conditions, e.g.:
  - Type of Service
  - User Class
  - Higher-level protocol, etc.

Visa Distribution After entry authorization is obtained and the new visa is issued, ACS_b forwards the VISA-GRANT packet to the requesting ACS_a. A VISA-GRANT also has to be sent to GW_b so it can process entering packets accordingly. However, in the original VISA-GRANT packet, S_b^a is encrypted with EK_{ACS_a} to maintain its secrecy. Since DK_{ACS_a} is not known to GW_b, ACS_b has to create a different copy of a VISA-GRANT for delivery to GW_b

where S_b^a is encrypted with EK_{ACS_b}. Upon receipt of the VISA-GRANT, GW_b creates a new entry in its visa-table.

When ACS_a receives a VISA-GRANT packet, it has to check several conditions:

- First, ACS_a verifies that H_a, H_b and TS_a are indeed the same values that were used in the VISA-REQUEST packet.
- Next, the integrity of the VISA-GRANT packet has to be verified. ACS_a can do so by re-computing the packet signature with EK_{ACS_b}.[4]
- Finally, the authentication type and the expiration conditions are checked for recognizability and consistency.

When the VISA-GRANT packet is validated, ACS_a notifies its visa-router, GW_a via a VISA-GRANT packet, where S_b^a is encrypted with EK_{ACS_a}[5] and the entire packet is signed with DK_{ACS_a} instead of DK_{ACS_b}. On receipt of the VISA-GRANT, GW_a installs the new entry in its visa-table and becomes ready to pass packets between H_a and H_b. The VISA-GRANT packet sent to H_a is similar save for S_b^a which is encrypted with K_{H_a}.

Setup Summary In summary, the setup phase involves the following steps:[6]

1. $H_a \Longrightarrow ACS_a : HOST - REQUEST$
2. $ACS_a \Longrightarrow ACS_b : VISA - REQUEST$
3. $ACS_b \Longrightarrow GW_b : VISA - GRANT^{GW_b}$
4. $ACS_b \Longrightarrow ACS_a : VISA - GRANT^{ACS_a}$
5. $ACS_a \Longrightarrow GW_a : VISA - GRANT^{GW_a}$
6. $ACS_a \Longrightarrow H_a : VISA - GRANT^{H_a}$

In the description above, $VISA - GRANT^P$ denotes the *Visa* packet where S_b^a is encrypted for the principal P (e.g., GW_a) with a key which is either P's public key, or a key shared among the sender and P. All other fields are the same for all $VISA - GRANT^P$-s.

Of all the messages exchanged during the setup phase, those in steps (2) and (4) are of particular importance since they cross AD boundaries and are, therefore, subject to much greater exposure.

5.2 Packet Forwarding

When the setup phase is completed, H_a can begin communication. Each packet that H_a sends to H_b has to be accompanied by a visa-stamp. However, it is not enough to simply sign the packet. The reason for this is twofold. First, the protocol may need to support multiple visas for the same end-system pair. Therefore, a visa-router can not uniquely identify an entry in its table using only

[4] Freshness and uniqueness of the VISA-GRANT packet is evident from the presence in it of TS_a.

[5] It is assumed that GW_a and ACS_a share the same public/private key pair.

[6] The notation $A \Longrightarrow B : PACKET$ means *"A sends PACKET to B "*.

the two end-system addresses. For this reason, TS_a and TS_b are included with each packet. Second, since one of the protocol goals is the detection of *stale*, (or potentially replayed) packets, H_a needs to attach a timestamp to each packet. In this context, a regular (non-visa) packet is composed of a data segment and a network-layer header.

A visa-stamp is computed as:

$$VISA - STAMP = F([HEADER, DATA, TS_{H_a}], S_b^a) \tag{5}$$

where F is the signature function determined by the *auth-type* agreed to during the setup phase and TS_{H_a} is the timestamp assigned to the packet by H_a. TS_{H_a} must be unique, i.e., no two packets should carry the same timestamp (for a given connection).

The resulting data packet has the following form (see also Figure 2):

$$DATA - PACKET = [HEADER, DATA, VISA - STAMP, TS_a] \tag{6}$$

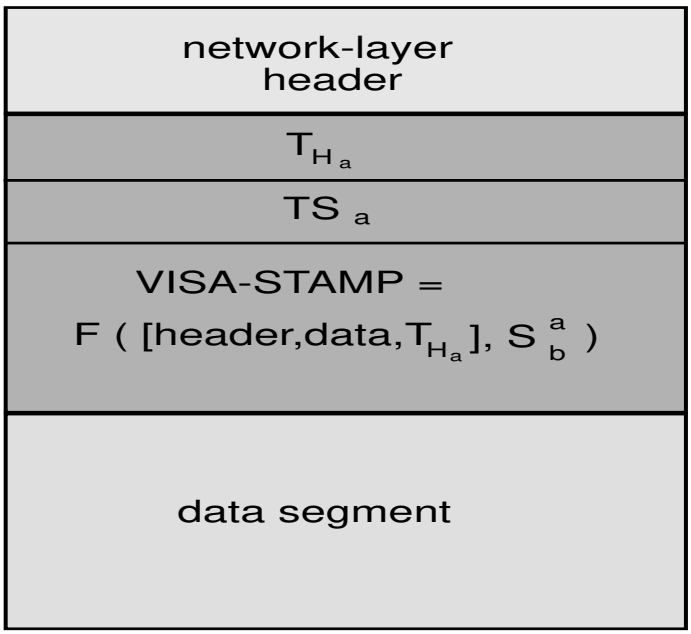

Fig. 2. Data packet format

Exiting AD_a As described in Section 2, when a packet with an attached visa-stamp arrives at GW_a it has to demonstrate authorization to leave AD_a as well as authenticity and freshness of its contents.

GW_a checks the first condition by indexing its visa-table with the $[TS_a, H_a, H_b]$ tuple. Successful look-up indicates the existence of exit authorization by ACS_a. Next, the freshness of the packet is verified by comparing the packet timestamp, TS_{H_a} with the stored timestamp of the last (previous) data packet that used the same visa (this value is referred to as $T_{last}^{(a,b)}$). Finally, GW_a re-computes the packet signature and compares it to the visa-stamp attached to the packet. If the two values match, GW_a can safely conclude that the packet contents are authentic.

The order in which these checks are performed is not arbitrary. In particular, the reason for verifying packet freshness before validating a packet signature has to do with the cost of the latter operation. Naturally, for *authentic* packets, the order of these two operations is immaterial. However, because we would like to detect replayed (i.e., *stale*) packets at the earliest possible time, and because it is significantly cheaper to compare two timestamps than to re-compute a packet signature, the freshness test is performed first.

Entering AD_b The goals of GW_b with respect to entering packet traffic are similar. Like GW_a, GW_b checks authorization by indexing its visa-table with the $[TS_b, H_a, H_b]$, verifies packet freshness by comparing the packet timestamp with the its $T_{last}^{(a,b)}$, and re-computes the signature to verify the authenticity of the packet contents.

5.3 Teardown

As mentioned previously, a visa may be terminated for one of two reasons:

- Normal expiration (time, packet or data ceiling reached)
- Explicit revocation (by explicit order of an ACS)

In the first case, a visa-router simply deletes the corresponding entry from its visa-table. If and when a packet bearing a visa-stamp computed using an expired visa arrives at the visa-router, it is promptly discarded since the table look-up fails.

In case of explicit revocation, an ACS may decide for some reason that a certain visa is no longer trusted and sends a REVOKE packet to the appropriate visa-router and a peer ACS:

$$REVOKE = [H_a, H_b, TS_a, TS_b, Reason]^{DK_{ACS}} \tag{7}$$

TS_a and TS_b in the REVOKE are the same visa identifiers assigned at the setup time. Since a visa can only be revoked once, there is no need to timestamp a REVOKE packet as its replay presents no danger.

6 Design Issues

In this section, we discuss several issues leading to the design of new *Visa* protocol. We also address several important features where the current protocol differs from its predecessors.

6.1 Visas

In the previous *Visa* protocol versions, each end-point AD issued a visa for authorized communication. Moreover, each visa had an associated visa-key, thereby necessitating the computation and subsequent verification of two distinct packet signatures. In other words, the end-system computed two packet signatures, $DSIG_{exit}$ and $DSIG_{entr}$. The former was verified by the exit visa-router in the source AD, and the latter, by the entry visa-router in the destination AD.

The setup phase of the protocol consisted of the following steps:

1. $H_a \Longrightarrow ACS_a : HOST - REQUEST$
2. $ACS_a \Longrightarrow ACS_b : VISA - REQUEST$
3. $ACS_b \Longrightarrow GW_b : VISA - GRANT_{entr}$
4. $ACS_b \Longrightarrow ACS_a : VISA - GRANT_{entr}$
5. $ACS_a \Longrightarrow GW_a : VISA - GRANT_{exit}$
6. $ACS_a \Longrightarrow H_a : VISA - GRANT_{entr}, VISA - GRANT_{exit}$

The primary reason for each ACS issuing its own visa (one entry and one exit) was the assumed lack of trust between the two ACSs as far as the issuance of *good* keys or visas. On the other hand, ACS_a still *trusts* its counterpart, ACS_b, enough not to *misuse* $VISA-GRANT_{entr}$ (and the included key). In other words, ACS_b issues a visa which it subsequently releases to ACS_a for distribution to H_a. This trust in ACS_a seems to contradict the reason for issuing two visas.

In the protocol described in Section 5.1, only one visa-key and a single visa is used for both exit of AD_a and entry of AD_b.

6.2 Replay Prevention

As motivated by our arguments in Chapter 1, one of the goals of an effective stub policy enforcement mechanism is the protection of stub AD network resources from unauthorized traffic. This includes traffic flowing to and from *equipped* end-systems. Previous *Visa* protocol incarnations [5] dealt with this issue by ensuring data integrity and authenticity of the packets crossing AD boundaries. One deficiency in the previous design was the absence of any provisions for detecting replayed packets belonging to authorized connections. This has been remedied (as evidenced in Section 5) by requiring that end-systems attach a unique timestamp to each visa-stamped packet and visa-routers keep a per visa record of a timestamp for the last packet processed.

One important insight related to this requirement is the apparent impossibility of effective replay detection without state in visa-routers. With some degree of synchronization, a stateless visa-router may be able to detect *very old* packets. However, if an intruder duplicates each legitimate packet and injects duplicates into the packet stream shortly after the corresponding original packets, duplicates can not be detected.

6.3 Visa Expiration

The only method for visas to expire in the previous protocol versions is by way of timeouts, i.e., an explicit time limit is negotiated at setup time and a visa is invalidated when the time limit is exceeded. While this is adequate for some types of connections, provisions for other methods of visa expiration may be necessary. These include limits on inactivity periods, number of packets and bulk data transferred.

Unfortunately, expiration based on limits other than just simple timeouts is not possible without state in visa-routers. In order to expire visas according to any of the above criteria requires that a visa-router maintain running tally of packets, data bytes or the time of last packet arrival on a per visa basis.

6.4 Visa Revocation

Visa termination by explicit order from an ACS should be viewed as more of exception than the norm as, ordinarily, visas are terminated by exceeding some limit negotiated at the time of issuance. Nonetheless, in circumstances where there is suspicion of a visa's compromise an ACS may need to revoke a visa prematurely, i.e., before its resource limits are reached.

In order to revoke an active visa, an ACS contacts the appropriate visa-router and identifies the visa targeted for termination. Thereafter, the visa-router has to ensure that no more packet traffic belonging to the revoked visa connection passes through. In a stateful model, a visa-router can simply delete the entry from its table thereby barring any further traffic. In a completely stateless model where each packet carries the entire visa along with the derived visa-stamp, a visa-router has to keep **state** with respect to the revoked visa. (Otherwise, it can not distinguish *bona fide* visas from revoked ones). A direct consequence of this requirement is the inability of a completely stateless visa-router to support visa revocation.

6.5 Coverage of Packet Signatures

In the context of this section, a packet is composed of two portions: network-layer header and data. As discussed in Section 2, data authenticity is one of our primary goals. Hence, there is no question as to whether or not the data portion needs to covered by the packet signature. Network header is a different matter, however. A typical network-layer header contains addressing information such as source and destination end-system addresses, packet sequence number, packet length and other fields. (Figure 3 depicts the IP [21] datagram header, for example).

Any header field not covered by the packet signature leaves a potential covert channel, since an intruder could trap a valid packet, change the unchecked field, and forward the modified copy without raising suspicion. We could protect against this by including the entire network-layer header under the packet signature, but in most internetworking protocols there are some header fields that

are modified by the intermediate routers, and hence cannot be included in the signature. (All routers may have to modify the header, not just visa-routers, and we assume that non-visa routers do no regenerate the signature. If a public-key method is used, not even visa-routers can do so.)

For example, there are two such variable fields in the IP protocol. One is the header checksum; this cannot be forged because it is a function of the other fields in the header, and is already recomputed by each IP router. The other is the 8-bit *Time-To-Live* (TTL) field, used to prevent packet from looping forever. The TTL must be decremented by each IP router, and must never be incremented. An intruder could communicate approximately 6 or 7 bits per datagram by manipulating the initial value of the TTL field in copies of otherwise validly-signed packets.

If this covert channel is a reason for concern, there are a number of steps that can be taken. The entry visa-router (GW_b) can use the knowledge of its AD topology to reduce the TTL value to the minimum necessary for the packet to safely arrive at H_b, thereby reducing the bandwidth of this covert channel. Alternatively, GW_b could always set the TTL to its maximum value. (However, this would violate the letter of the IP specification, and might confound protocols that use the TTL field to limit the lifetime of a packet).

Another issue has to do with the addressing information in the network header. Recall that visas are issued on the basis of the source and destination end-system addresses (sometimes, along with the transport-layer protocol number). As described in Section 5.2, each visa-related data packet carries a visa identifier (TS_a). This identifier is stored alongside the two end-system addresses in visa-tables of both GW_a and GW_b. Since a visa-router still has to consult its visa-table to look up the visa-key, it can (inexpensively) verify the H_a, H_b addresses as well. Therefore, it can be argued that end-system addresses and other information (e.g., type-of-service, transport protocol, etc.) that is stored in the visa-table entry does not need to be protected by the packet signature.

6.6 Fragmentation

In a number of internetworking protocols (e.g., IP) a router may have to fragment a packet if it cannot be transmitted in a single unit. Data signatures complicate the use of fragmentation since the fragments must appear to have been signed by H_{src}, but the signatures would have to be computed by the fragmenting router. With signatures based on conventional cryptography, fragmentation is a problem because only a visa-router can do it while preserving the data signatures. With public-key signatures, this is impossible, since only the originating end-system can compute the signature.

Fragmentation is at best a necessary evil [12]; it is almost always better to set packet sizes at H_{src}, to make the best possible use of the available bandwidth and to provide acknowledgements for each transmission unit. Although methods have been proposed for accommodating fragmentation while preserving data signatures [26], we insist that the source end-system avoid sending packets that will have to be fragmented. A router should assist in this by returning an error

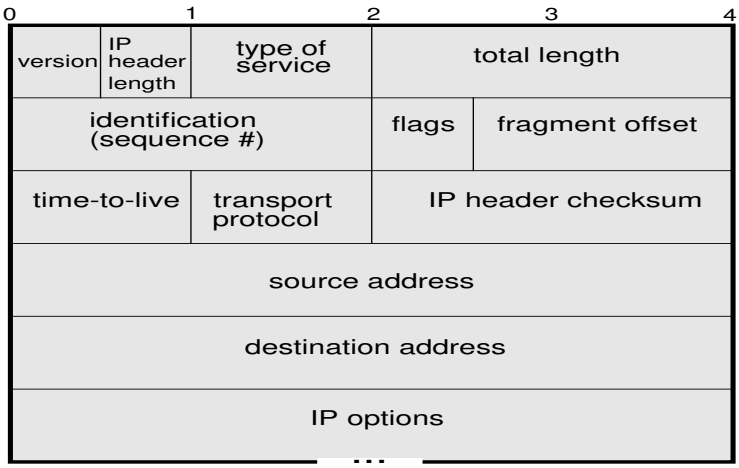

Fig. 3. IP datagram header

packet when it is unable to transmit a data packet without fragmenting it; in fact, the IP protocol includes a mechanism for doing so through the use of ICMP *Destination Unreachable/Fragmentation Needed* message [22].

6.7 Loss of State

Reliable state in visa-routers (i.e., visa-tables) is one of the fundamental features and requirements of *Visa* protocol presented in this paper. Nevertheless, it is unreasonable to expect visa-routers to be non-faulty at all times. The implication is that loss of state in visa-routers may occur infrequently and provisions must be made for the reinstatement of state after when a visa-router becomes operational. For this reason, an initializing visa-router sends a special ROUTER-UP packet to its local ACS. (Visa-routers are required to keep local ACS address(es) in stable storage). Upon receipt of a ROUTER-UP, ACS replies with one or more VISA-LIST packets which contain a set of all currently valid visas issued to the visa-router in question. The format of a VISA-LIST is similar to that of a *Visa* packet, except that it contains multiple visa records. ROUTER-UP only specifies the identity of the initializing visa-router.[7]

[7] An intruder masquerading as a legitimate visa-router may generate ROUTER-UP packets, however, visas contained in the subsequent VISA-LIST are only intelligible to genuine visa-routers.

In spite of the mechanism described herein, not all state can be effectively recovered. In particular, visa expiration conditions such as inactivity timeouts, data and packet limits require maintaining state (i.e., usage meters) that is updated on a per packet basis. Unless it is kept in stable storage (an unreasonable requirement), this state is irrecoverably lost when a visa-router fails.

6.8 Stateful Model

In this section, we summarize the reasons for selecting stateful over stateless visa-router model. There are several incentives for maintaining state in visa-routers:

- Replay prevention
  As discussed in Section 6.2, it is impossible to detect replayed packets in a visa-router without maintaining state. This is best illustrated by a situation whereby an intruder simply duplicates each valid packet and forwards it immediately after the original. Because the two packets are *back-to-back*, even if GW_a has some idea of the $H_a \rightarrow GW_a$ delay, it would accept both packets as valid.
- Visa expiration by means other than timeouts
  In order to expire visas by means of data, packet or idle-time limits, a visa-router must maintain running tallies. This is impossible to achieve with a stateless router model.
- Visa revocation
  If an ACS decides that it no longer trusts a previously approved connection, it may need to revoke a visa prematurely. As described in Section 6.4, ACS explicitly notifies the appropriate visa-router (via a REVOKE packet) that a certain visa is no longer valid. To uphold the revocation of a visa, i.e., to let no more packets bearing derived visa-stamps through, the visa-router has to maintain a record (state) with respect to the revoked visa.
- Interplay with transit policy enforcement
  A final reason has to do with the integration of stub and transit policy enforcement mechanisms. As described in [7, 28], transit policy enforcement in stub ADs also takes place at AD boundaries in entities referred to as *Policy Gateways* (PGs). As it turns out, each PG in a stub AD maintains a table of end-system pairs that are engaged in inter-AD communication. The purpose of this table is to map end-system pairs into so-called *Policy Routes*. Although a PG and a visa-router are logically distinct, for the most part, they share the same physical location.

7 Security Analysis

In this section we address the security of *Visa* protocol presented earlier in the paper. As mentioned before, intra-AD messages and other intra-AD communication is assumed to be secure and each AD is assumed to employ an authentication mechanism of sufficient strength. What remains to be shown is the security of

the inter-AD communication, i.e., the two messages exchanged among ACS_a and ACS_b as part of the setup phase and the subsequent data packets.

For each of these messages, two security issues are of interest: i) whether or not the message conveys the information necessary to establish its origin, authenticate its contents and assure freshness, and ii) whether it can be used maliciously (e.g., if it is intercepted by an intruder) to achieve unauthorized or otherwise compromised communication.

7.1 VISA-REQUEST

The first inter-AD message in the setup phase is the VISA-REQUEST:

$$ACS_a \Longrightarrow ACS_b : VISA - REQUEST \tag{8}$$

$$VISA - REQUEST \ = \ [\ H_a, H_b, TS_a\ ]^{\,DK_{ACS_a}} \tag{9}$$

Upon its arrival, ACS_b has to verify that ACS_a originated **this exact** packet **recently**. Once ACS_b obtains $CERT_{ACS_a}$, extracts from it EK_{ACS_a} and recomputes the packet signature it is assured that the packet is originated by ACS_a and is authentic. Of course, this is very much dependent upon the strength of the signature mechanism.

Both timeliness and uniqueness of the VISA-REQUEST are established by examining the TS_a field. Recall that TS_a is the timestamp assigned by ACS_a. ACS_a is responsible for ensuring that it has never been used before. As described in [14], ACSs's clocks are not necessarily closely synchronized, i.e., a certain clock skew is expected. We assume, there exists an upper bound (referred to as Δ_T) on the clock skew between any two ACSs. By comparing its current clock reading to TS_a, ACS_b can establish the timeliness of the VISA-REQUEST. However, *timeliness* is relative to Δ_T since an intruder can still delay a VISA-REQUEST for, at most, the value of Δ_T.

Uniqueness is a different matter. In order to establish that a VISA-REQUEST has not been *seen* before, each ACS has to keep state in the form of a *peer-table*. Each entry in the peer-table corresponds to a peer ACS (in a different AD) which has previously communicated with the ACS in question. Among other information (such as public key certificates), each entry stores the timestamp of the last VISA-REQUEST by the corresponding ACS. (We refer to this value as TS_a^{last}). If $TS_a < TS_a^{last}$, ACS_b can suspect a replay attack. However, it can not be absolutely sure since VISA-REQUEST packets can arrive out of order. If $TS_a = TS_a^{last}$, the VISA-REQUEST packet is obviously a duplicate.

As far as the intruder is concerned, there is little value in a VISA-REQUEST packet. It contains no secret fields and it only reveals that a visa for communication between H_a and H_b has been requested (hence, a VISA-GRANT may flow in the opposite direction soon thereafter). Also, modification of packet data is *unproductive* as each VISA-REQUEST is protected by an unforgeable signature. Replay of a VISA-REQUEST is just as futile as can be seen from the above discussion.

Another security-related aspect is the issue of the implied beliefs carried in a VISA-REQUEST. Specifically, it conveys to ACS_b that i) ACS_a authorizes

the $[H_a, H_b]$ communication, and ii) believes that the requesting end-system H_a is authentic. ACS_b has no reason to question or doubt the former (since it is subject to AD_a's local policy embodied in ACS_a), however, with regard to authentication of H_a, ACS_b has to *take ACS_a's word for it.* In other words, the end-result is that: *ACS_b believes that ACS_a believes that H_a is authenticated.* This conclusion may be too weak in some circumstances. In order to obtain a stronger conclusion such as *ACS_b believes that H_a is authenticated*, either:

1. ACS_b has to conduct a separate, possibly higher-level, authentication dialog with H_a, or
2. ACS_a has to include an unforgeable proof of identity for H_a as part of the VISA-REQUEST

The second choice is preferable because it does not involve relying on other protocols and introduces no additional communication overhead. However, it has two important drawbacks: i) the problem of scale with regard to issuing proof-of-identity certificates at the granularity of end-systems, and ii) the additional overhead incurred by the use of public-key encryption for subsequent communication. This subject is discussed further in Section 7.3 below.

7.2 VISA-GRANT

VISA-GRANT is the second inter-AD message exchanged as part of the setup phase.

$$ACS_b \Longrightarrow ACS_a : VISA - GRANT^{ACS_a} \tag{10}$$

$$VISA - GRANT = [H_a, H_b, TS_a, (S_b^a)^{EK_{ACS_a}}]^{DK_{ACS_b}} \tag{11}$$

When ACS_a receives a VISA-GRANT packet, it has to:

- match the TS_a found in the packet with the corresponding TS_a stored in its table of outstanding VISA-REQUESTs. A successful match, in itself, demonstrates both freshness and uniqueness of the VISA-GRANT packet.
- authenticate the contents of the packet. Since ACS_a already knows EK_{ACS_b}, it can re-compute the packet signature and establish that ACS_b originated this packet and its contents are authentic.

As with a VISA-REQUEST, an intruder can capture a valid VISA-GRANT packet and attempt to attack the protocol. However, since a VISA-GRANT is signed and timestamped, no modification or replay is possible without detection. A danger even graver than successful fabrication or replay of a VISA-GRANT packet is the possibility of an intruder discovering S_b^a. (This would permit the intruder to generate arbitrary "genuine" packets). Therefore, the effectiveness of *Visa* protocol depends to a great extent on the strength of the underlying public-key encryption (signature) function.

7.3 Data packets

Recall that data packets belonging to a visa connection have the following structure:

$$DATA-PACKET = [HEADER, DATA, VISA-STAMP, TS_a] \qquad (12)$$

A data packet arriving at GW_a or GW_b, has to show that:

- it originated at H_a and is addressed to H_b
- it was sent recently
- it was not modified in transit

We begin by observing that the method by which GW_a determines the origin of a data packet is the verification of the visa-stamp attached to the packet which is computed with a secret visa-key, S_b^a. In addition to H_a, S_b^a is known to four other principals: H_a, ACS_a, GW_a, ACS_b and GW_b. This implies that any of these principals can potentially generate *genuine* visa-stamps.

After verifying the visa-stamp, timestamp and addressing information of a data packet, the only conclusion that GW_a can make is: *the packet contents are authentic, the packet is not a replay, and its origin can be any of* H_a, GW_b *or* ACS_b. We assume that GW_a believes in its own *goodness*, thus it can be sure that it did not originate the said data packet, and, GW_a trusts ACS_a enough to believe that ACS_a did not originate the packet. Similarly, GW_b can establish that the origin of a data packet can be H_a, GW_a or ACS_a.[8]

The uncertainty about the exact origin of data packets is, potentially, a reason for concern. Nonetheless, we argue that the overall security and robustness of *Visa* protocol relies on the invulnerability of the ACSs and visa-routers. If the security of an ACS or a visa-router is somehow compromised, fraudulent visas may be issued and unauthorized communication can transpire. On the other hand, it is worth considering, if only for academic reasons, what it would take to remove the above uncertainty, i.e., allow visa-routers to determine the source of each data packet unambiguously.

To do so would require that each end-system be able to certify its identity, i.e., be uniquely identifiable by a some property that can be used to sign data leaving no doubt about the identity of the signature creator. The immediate consequence of this statement is that conventional cryptography is unable to solve the problem since it calls for (at least) two principals sharing a key. (If ACS_a and H_a possess the same key, K_{H_a}, any one of them can generate data signatures with that key).

The only remaining choice is the use of public-key cryptography. Suppose that each participating end-system is issued a public-key pair $[EK_{H_a}, DK_{H_a}]$ and a corresponding certificate, $CERT_{H_a}$, by a well-known authority.[9] Instead of issuing a secret, conventional visa-key, ACS_b would generate a public-key pair, $[EK_b^a, DK_b^a]$, and a VISA-GRANT packet would take the form of:

$$VISA-GRANT = [H_a, H_b, TS_a, , EK_b^a, (DK_b^a)^{EK_{H_a}}]^{DK_{ACS_b}} \qquad (13)$$

[8] Note that if the protocol operates correctly, none of GW_a, GW_b, ACS_a and ACS_b ever generate data packets signed with S_b^a.

[9] The authority can not be ACS_a or any other entity in AD_a, for obvious reasons.

Note that S_b^a is replaced by DK_b^a, the secret (signature) component of the public-key pair. Because it is encrypted with EK_{H_a}, DK_b^a can only be computed by H_a. However, anyone (ACS_a, GW_a, GW_b, etc.) can obtain the corresponding public key, EK_b^a. If H_a generates data packet visa-stamps with DK_b^a, both GW_a and GW_b can trace the packet source to H_a as no other entity can be in possession of DK_b^a.

While this approach results in somewhat increased protocol security, its benefits are outweighed by the cost of using public key encryption on a per packet basis. Another drawback is (as alluded to in Section 7.1) the issue of scale with respect to certification of the individual end-systems. Finally, one of the fundamental protocol assumptions is the security of the ACSs and visa-routers. Because ACSs are presumed to exercise extensive control over their constituent end-systems, it may be unreasonable to trust an end-system in an AD whose ACS or visa-router is not trusted.

8 Protocol Costs

In this section we address the impact of *Visa* protocol on its participants. In particular, we consider the costs incurred per visa as well as per data packet.

8.1 Setup and Distribution

The setup phase involves at least two packets: a HOST-REQUEST from H_a to ACS_a, and a VISA-REQUEST from ACS_a to ACS_b. The distribution phase involves at least four more packets: $VISA-GRANT^{ACS_a}$, $VISA-GRANT^{GW_b}$, $VISA-GRANT^{GW_a}$ and $VISA-GRANT^{H_a}$.[10] In total, a minimum of six packets are exchanged, yet only two of the six travel across AD boundaries.

8.2 State Overhead

State overhead, consisting of storage costs, is introduced in this protocol mainly by the need for all participants, but especially visa-routers, to keep visa-tables.

In order to facilitate fast packet switching, routers often use specialized hardware equipped with very fast memory. However, this memory is expensive and, hence its availability is limited. Therefore, storage overhead in visa-routers is of particular importance. Its major contributing factor is the maintenance of a visa-table where each entry corresponds to a currently valid visa. Each entry contains essentially the same information as carried in the corresponding VISA-GRANT packet. The only exception is that visa-routers must constantly monitor bandwidth usage of individual connections. Also, the timestamp of the last data packet, $TS_{last}^{(a,b)}$ is kept in order to i) prevent replay of old packets, and ii) monitor the inactivity time.

[10] Additional exchanges may be required in order for H_a to authenticate itself to ACS_a and ACS_b.

An ACS must also keep a table of active visas. ACS's visa-table is essentially the same as the visa-router's visa-table, except ACSs are not required to keep running counters. In addition, an ACS needs to keep a *peer ACS* table which contains certificates for ACSs in other ADs and, for each peer ACS, a timestamp of the last VISA-REQUEST received from that ACS.[11]

8.3 Per packet costs

The per packet costs include the additional fields in data packets, table look-ups in visa-routers, and data signature computations.

Each data packet carries a visa header depicted in Figure 2 above. The visa identifier, TS_a is 64-bit quantity (sufficient width for a timestamp) and the end-system timestamp, TS_{H_a}, is also 64 bits wide. The size of the visa-stamp depends on the signature method, e.g., 128 bits for MD4 secret prefix/suffix or 64 bits for DES-based MAC.

The overhead due to data signature operations depends upon the particular signature scheme used. It also depends upon whether the operation involves passing over the entire or only part of, the packet. Furthermore, the cost of signature computation can be significantly more expensive than the cost of signature verification as in some proposed variations of RSA [24]. In general, there are three operations involved: signature computation at H_a and two signature verifications at GW_a and GW_b, respectively.

9 Summary

In conclusion, the new *Visa* protocol presented in this paper is a simple and powerful mechanism for controlling packet traffic at stub AD network boundaries. The primary goal of *Visa* protocol, i.e., protection of stub AD network and end-system resources from unauthorized access, is achieved by requiring that all communication be first authorized by the ACSs in end-point ADs. Authorization to communicate is then embodied in a visa which is issued to the requesting end-system and distributed to the intervening visa-routers. Subsequent packets carry unforgeable visa-stamps which are verified by the visa-routers. Visas are terminated when the underlying communication exceeds a pre-determined resource limit.

Visa protocol does not impact intra-AD communication and has no influence on the end-systems that do not partake in inter-AD communication. Although the cost of connection setup is somewhat high, per packet overhead due to the protocol is reasonable, especially, when non-cryptographic packet signatures are used.

[11] See Section 7.1.

References

1. M. Burrows, M. Abadi, and R. Needham, *A Logic of Authentication*, **Proceedings of ACM Symposium on Operating System Principles**, December 1989.
2. W. Diffie, *The First Ten Years of Public-Key Cryptography*, **Proceedings of the IEEE**, Vol. 76, No. 5, May 1988.
3. D. Clark, *Policy Routing in Internet Protocols*, **Journal of Internetworking: Research and Experience**, October 1990.
4. W. Diffie and M. Hellman, *New Directions in Cryptography*, **IEEE Transactions on Information Theory**, Vol. 22, No. 11, November, 1976.
5. D. Estrin, J. Mogul, G. Tsudik, *Visa Protocols for Controlling Inter-Organizational Datagram Flow*, **IEEE Journal on Selected Areas in Communications**, May 1989.
6. D. Estrin and G. Tsudik, *Secure Control of Transit Internetwork Traffic*, **Computer Networks and ISDN Systems**, October 1991.
7. D. Estrin and G. Tsudik, End-to-End *Argument for Network-Layer Access Controls*, **Journal of Internetworking: Research and Experience**, June 1991.
8. D. Estrin, *Policy Requirements for Inter Administrative Domain Routing*, **Computer Networks and ISDN Systems**, June 1991.
9. J. Galvin, K. McCloghrie, J. Davin, *Secure Management of SNMP Networks*, **Proceedings of 1991 Integrated Network Management Symposium**, April 1991.
10. International Standards Organization, *Information Processing Systems – Open Systems Interconnection – Basic Reference Model*, **ISO 7498**, 1977
11. International Standards Organization, *Security Architecture*, **ISO 7498-2-1988(E)**, 1988
12. C. Kent and J. Mogul, *Fragmentation Considered Harmful*, **Proceedings of 1987 ACM SIGCOMM Symposium**, August 1987.
13. J. Linn, *Privacy Enhancement for Internet Electronic Mail. Part I: Message Encipherment and Authentication Procedures*, **RFC 1113, SRI Network Information Center**, January 1989.
14. B. Liskov, L. Shrira and J. Wroclawski, *Efficient At-Most-Once Messages Based on Synchronized Clocks*, **ACM Transactions on Computer Systems**, MAy 1991.
15. P. Mockapetris, *Domain Names - Implementation and Specification*, **RFC 1035, SRI Network Information Center**, November 1987.
16. J. Mogul, *Simple and Flexible Datagram Access Controls for Unix-based Gateways*, **Proceedings of Summer 1989 USENIX Technical Conference**, August 1989.
17. National Bureau of Standards, *Federal Information Processing Standards*, National Bureau of Standards, Publication 46, 1977.
18. R. Needham and M. Schroeder, *Using Encryption for Authentication in Large Networks of Computers*, **Communications of the ACM**, Vol. 21, No. 12, December 1978.
19. R. Needham and M. Schroeder, *Authentication Revisited*, **ACM Operating Systems Review**, Vol. 21, No. 7, January 1987.
20. M. Padlipsky, *The Elements of Networking Style*, **Englewood Cliffs, NJ:Prentice Hall**, 1985.

21. J. Postel, *Internet Protocol*, **RFC 791, SRI Network Information Center**, September 1981.
22. J. Postel, *Internet Control Message Protocol*, **RFC 792, SRI Network Information Center**, September 1981.
23. R. Rivest, *The MD4 Message Digest Algorithm*, **Proceedings of CRYPTO'90**, August 1990.
24. R. Rivest, A. Shamir and L. Adleman, *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, **Communications of the ACM**, February 1978.
25. J. Steiner, C. Neuman, J. Schiller, *Kerberos: An Authentication Service for Open Network Systems*, Proceedings of USENIX Winter Conference, February 1988.
26. G. Tsudik, *Datagram Authentication in Internet Gateways*, **IEEE Journal on Selected Areas in Communications**, May 1989.
27. G. Tsudik, *Message Authentication with One-Way Hash Functions*, **Proceedings of IEEE Infocom 92**, May 1992.
28. G. Tsudik, *Access Control and Policy Enforcement in Internetworks*, **Ph.D. Dissertation, USC Computer Science TR-91-15**, April 1991.

A Experimental Results

A.1 Experimental Platform

For the purposes of functionality and performance testing *Visa* protocol was implemented within the SunOS 4.1 environment. Sun *SparcStation 1+* workstations were used as the hardware platform. (*SparcStation 1+* scores 15.8 MIPs on the well-known *Dhrystone benchmark*). All workstations were equipped with 8 to 12 Mbytes of main memory.

The workstations directly participating in the experiments were interconnected with Ethernet segments to form a topology depicted in Figure 5. However, a number of other workstations were used as *secondary*, non-essential nodes (i.e., visa-hosts).

In all experiments, MD4 was used as the integrity mechanism for all packet traffic. Data packet authentication was performed using the secret prefix method in conjunction with MD4 as described in [27].

Signatures based on public-key encryption were not implemented. The main reason for this is the apparent lack of fast public key encryption implementation. Unfortunately, existing state-of-the-art public-key software implementations are excruciatingly slow. Even the fastest RSA hardware can only attain a meager 2-4 Kbits/second throughput [2]. Faster, scaled-down variants of RSA have been proposed [13]. For example, Privacy-Enhanced Electronic Mail specifies an RSA variant where signature verification costs significantly less than signature computation. This is achieved by using a large (e.g., 512-bit) secret exponent in conjunction with a very small (e.g., 8-bit) public exponent. Although signature verification becomes much cheaper (on the order of milliseconds), the cost of signature computation remains quite high (e.g., 4-5 seconds for a 512-bit modulus) [13, 9].

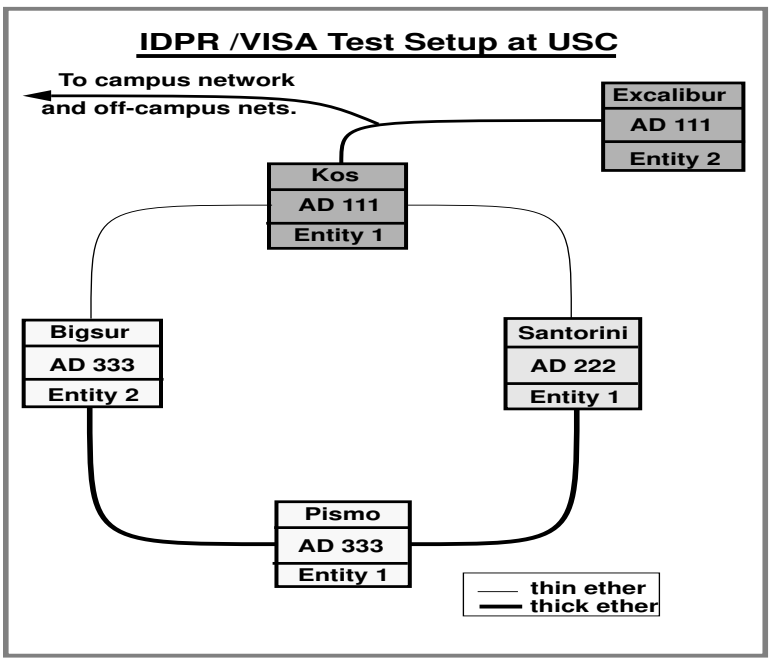

Fig. 4. Laboratory Test Environment

A.2 Visa Experiments

In its previous incarnation [5, 16], *Visa* protocol was implemented within the network layer which required modifications to IP. All visa-related information was carried within IP option fields. This had several unpleasant side-effects: i) the visa option fields exceeded 40 bytes which is the maximum IP option length, and ii) IP options were not transparent to IP routers (even those that do not implement *Visa* protocol). The former limits the length of visa-related data and control messages, while the latter contributes to increased processing delay at all (including non-visa) IP routers. In contrast to its predecessor, the current version is implemented as a self-contained protocol entailing no modifications to other protocol software.

All visa control packets are processed inside the kernel. This speeds up connection setup (as compared to out-of-kernel implementations [16]). Recall that connection setup is the most "overhead-intensive" phase of *Visa* protocol. It involves the exchange of at least two inter-AD packets: VISA-REQUEST from ACS_a to ACS_b and VISA-GRANT from ACS_b back to ACS_a. Moreover, at least four inter-AD packets are exchanged (one HOST-REQUEST, and three VISA-GRANTs).

For the purpose of measuring the overhead we define connection setup time

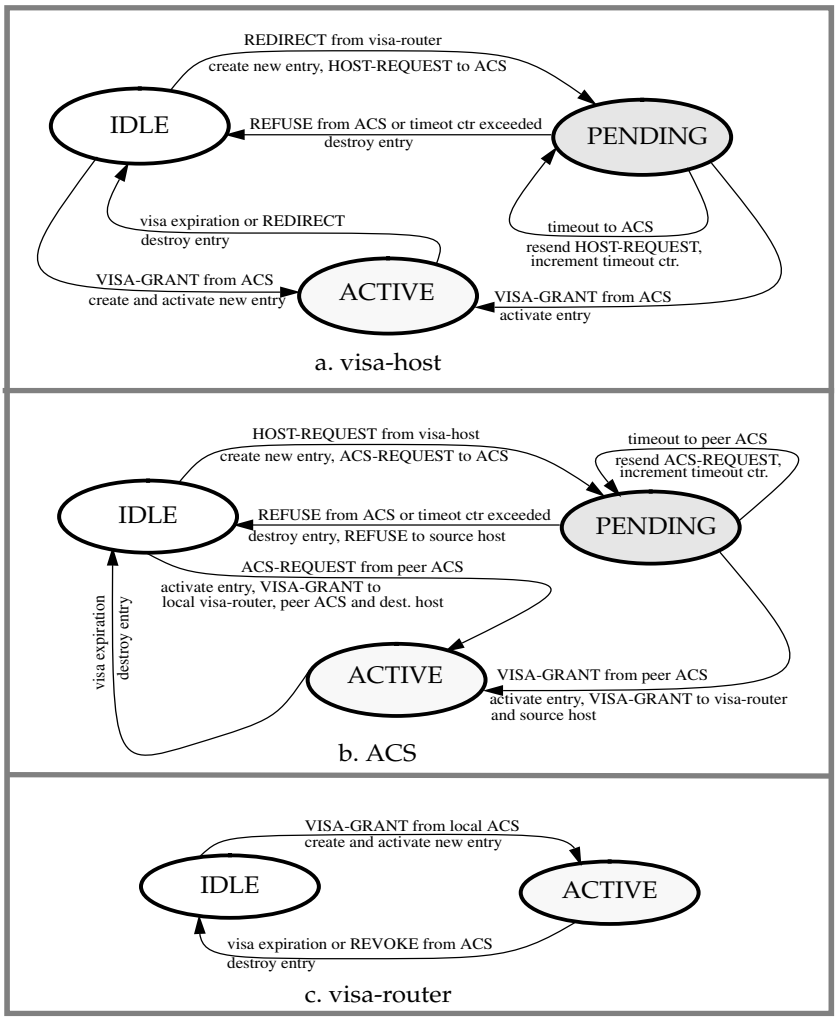

Fig. 5. Visa State Diagrams

as the interval between the arrival of a REDIRECT packet from a visa-router to the arrival of a corresponding VISA-GRANT from the ACS (as shown in Figure 6). We measured the setup time for directly connected ADs as well as for ADs separated by one transit AD hop. For the directly connected case, the timings ranged between 100 and 180ms (averaging around 130ms). In the case of one transit AD hop, the timings ranged between 110 and 200ms (averaging around 150ms). These results show that the overhead is still quite high. However, we expect that performance-tuning of the code should be able to cut these numbers down by about a third.

Encapsulation of data packets is performed by all protocol participants. Normally, all outbound data packets are passed from the transport protocol to IP where each packet is prepended with an IP header and handed over to the interface output function. *Visa* encapsulation is interposed between IP and the interface driver. (See Figure 7). This is done transparently so that neither IP nor any higher-layer protocol is aware of the encapsulation. Encapsulating a packet requires the following steps:

1. Visa-table lookup using $[SRC, DST, PROT, ToS]$ fields extracted from the IP header.[12]
2. Visa-stamp computation.

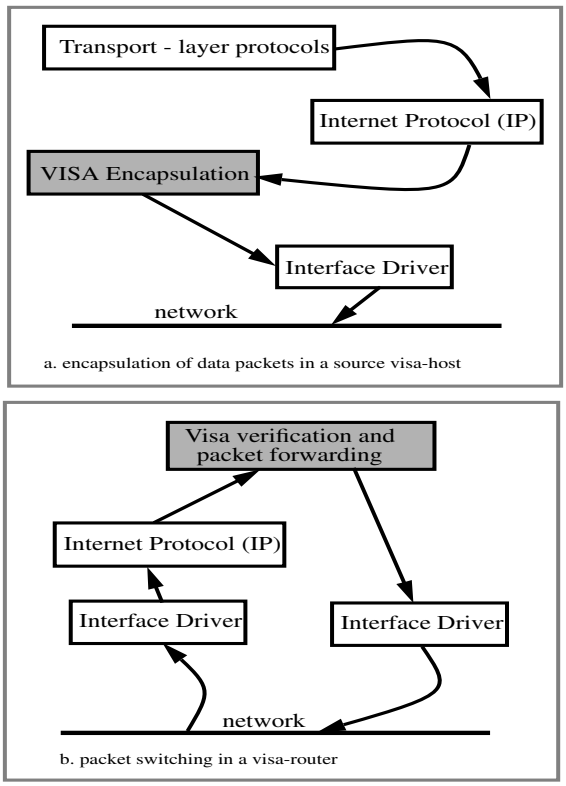

Fig. 6. Visa encapsulation and packet switching

[12] Unfortunately, this has to be done for all packets, even those that may not require a visa.

Packet switching at visa-routers involves the following operations:

1. Visa-table lookup with a visa identifier attached to each packet
2. Visa-stamp verification (i.e., integrity/authentication check)
3. *Light* bookkeeping, i.e., recording time of packet arrival, updating resource usage meters, etc.

We measured the overhead incurred by data packets flowing across previously established visa connections. The overhead measurements are derived using the ICMP Echo protocol [22]. In this protocol, a packet travels from the source to the destination end-system where it is immediately reflected back to the source. The results in Table 1 represent the the round-trip times (in milliseconds) for packets traveling between directly connected ADs. In other words, each packet travels across three ethernet segments: i) H_a to GW_a, ii) GW_a to GW_b, and, iii) GW_b to H_b where H_a refers to *pismo*, GW_a to *bigsur*, GW_b to *kos*, and, H_b to *excalibur* as depicted in Figure 5.

The first row shows the timings with *Visa* protocol inactive. The second row shows the timings for the same packet traffic, but, using *Visa* protocol without any integrity checking. These timings help us in identifying purely protocol-related overhead. The third row shows the results for *Visa* protocol with full integrity checking.

Packet size (bytes)	16	64	256	512	1024
No *Visa*	5	5	8	10	15
Visa w/o integrity	6	6	8.5	11	16
Visa with integrity	7.5	8	11	14	20

Table 1. Visa connection setup overhead (in ms)

First and foremost, these numbers clearly illustrate the relatively low cost of encapsulation and switching of visa packets. Another comforting aspect of our results is what appears to be insignificant additional overhead imposed by packet authentication and integrity checking. This may be due to the nature of MD4; since it is implemented entirely in software, data rates are constant independent of the packet size. (Hardware devices tend to offer much lower rates for small data units; mostly because of the fixed device initialization costs).

This article was processed using the LaTeX macro package with LLNCS style